AF
2700

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

#13
S. Cutts
4/6/04

In re Application of:
Sai V. Allavarpu
Rajeev Angal
Tony T. Vuong

§ Group Art Unit: 2124
§
§ Examiner: Chavis, John Q.
§
§ Atty. Dkt. No.: 5181-48500
§ P4505

Serial No. 09/553,970

Filed: April 21, 2000

For: Thread-Safe Remote Debugger

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

March 23, 2004
Date

Signature

APPEAL BRIEF

Mail Stop Appeal Brief
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed January 26, 2004, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054, as evidenced by the assignment recorded at Reel 010993, Frame 0892.

II. RELATED APPEALS AND INTERFERENCES

No other appeals or interferences are known which would directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1 – 15, 17 – 37, 39 – 58 and 60 – 64 are pending in the present application and are the subject of this appeal. Claims 1-7, 9-15, 17, 20-28, 30-37, 39, 42-50, 52-58, 60, and 63-64 stand finally rejected under 35 U.S.C. § 102(e). Claims 8, 18-19, 29, 40-41, 51, and 61-62 stand finally rejected under 35 U.S.C. § 103(a). A copy of Claims 1 – 15, 17 – 37, 39 – 58 and 60 - 64, as on appeal (incorporating all amendments), is included in the Appendix hereto.

IV. STATUS OF AMENDMEMNTS

No amendment to the claims has been filed subsequent to the final rejection. The Appendix hereto reflects the current state of the claims.

V. SUMMARY OF THE INVENTION

A thread-safe debugging system may include a thread-safe client debug service and a thread-safe dynamic remote control debug service, referred to as a remote control service, both of which reside on a client computer system with the client application to be debugged. *See* Fig. 3, and page 18, lines 7-11. The debug service and the remote control

service may provide application programming interfaces (APIs) which allow multi-threaded applications executing on the client computer system to take advantage of debug services in a thread-safe and dynamic manner. *See* page 22, lines 4-15. One or more debug objects may be associated with one or more components of the multi-threaded application. Each component may be responsible for a particular task or set of tasks, such as database access, interaction with users, interaction with a CORBA bus, etc. See Fig. 4, and page 20, lines 8-18.

The remote control service may provide a network protocol interface through which one may initiate and manage the debug services on the client computer system after initiation of the multi-threaded application. In other words, the debug services may be remotely and dynamically controlled. Debug services may be switched on or off after program execution has begun. *See* page 19, lines 4-14. In one embodiment, debug services may be switched on or off for a program running on the client computer system by issuing commands (e.g., control requests) from a third party application over a network. *See* page 21, lines 2-16.

In one embodiment, the remote control service may be operable to allow third party applications to switch the debug services on and off for each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects, such as by specifying the name of each desired debug object. The remote control service may also allow one to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name, such as with the use of character wild cards. This capability allows one to specify groups of debug objects to receive particular debug commands. *See* page 21, lines 2-16.

The debug services may include a thread-safe debug print function. In one embodiment, the debug print function is operable on the client computer system independently of the remote control service. The debug print function may provide debug

output for one or more threads of the multi-threaded application such that the debug output of each thread remains distinct from the debug output of the other threads. In other words, debug output from different threads may be presented such that the output is not interleaved or otherwise garbled. In one embodiment, the debug output of each of the one or more threads executing the multi-threaded application may be directed to an output target, such as a standard output terminal or a file. *See* page 19, lines 16-24. In one embodiment, the debug output may be directed to one or more recipient computer systems coupled to the client computer system. For example, the debug output may be directed to a diagnostic tool on a recipient computer system for analysis of problems encountered during the execution and debugging of the multi-threaded application. *See* page 23, lines 11-16.

Thread safety may be ensured throughout the debugging system through the use of thread-safe mechanisms such as locks. In one embodiment, for example, the debug output of each thread may be kept distinct by associating a mutex (mutual exclusion) lock with each output target. When a thread in the application invokes the print function to send the debug output to an output target, it may first acquire the associated mutex lock, then send the debug information to the output target, and finally, release the mutex lock. Since only one thread may acquire the mutex lock at a given time, other threads trying to acquire the mutex lock to send debug information to the output target must wait until that lock is released by the current thread owning the lock. *See* page 19, line 26 – page 20, line 4.

In one embodiment, the debug object class may be i18n-enabled to provide for debug output in a plurality of languages. The term "i18n" refers to internationalization, which is the process of designing an application so that it can be adapted to different languages and regions without requiring engineering changes. *See* page 23, lines 18-24.

VI. ISSUES

1. Whether claims 1-7, 9-15, 17, 20-28, 30-37, 39, 42-50, 52-58, 60 and 63-64 are

anticipated by Wygodny et al. (U.S. Patent 6,282,701) under 35 U.S.C. § 102(e).

2. Whether claims 8, 29 and 51 are patentable under 35 U.S.C. § 103(a) over Wygodny et al. (U.S. Patent 6,282,701) in view of "Applicant choice of languages used to implement his invention."

3. Whether claims 18-19, 40-41 and 61-62 are patentable under 35 U.S.C. § 103(a) over Wygodny et al. (U.S. Patent 6,282,701) in view of Kaler (U.S. Patent 6,467,052).

VII. GROUPING OF CLAIMS

Claims 1, 9, 10, 13-15, 17-22, 30-32, 35-37, 39-44, 52, 53, 56-58, and 60-64 stand or fall together.

Claims 2-4, 23-25, and 45-47 stand or fall together.

Claims 5, 26, and 48 stand or fall together.

Claims 6, 27, and 49 stand or fall together.

Claims 7, 28, and 50 stand or fall together.

Claims 8, 29, and 51 stand or fall together.

Claims 10, 31, and 53 stand or fall together.

Claims 11, 33, and 54 stand or fall together.

Claims 12, 34, and 55 stand or fall together.

The above claim groupings are for purposes of this appeal only. The reasons why each group of claims is believed to be separately patentable are explained below in the Argument.

VIII. ARGUMENT

A. Claims 1, 9-10, 13-15, 17-20, 22, 30-32, 35-37, 39-42, 44, 52-53, 56-58, and 60- 63

The Examiner rejected claims 1, 9-10, 13-15, 17, 20, 22, 30-32, 35-37, 39, 42, 44, 52-53, 56-58, 60 and 63 under 35 U.S.C. § 102(e) as being anticipated by Wygodny (U.S. Patent 6,282,701) (hereinafter "Wygodny"). The Examiner rejected claims 18, 19, 40, 41, 61 and 62 as unpatentable under 35 U.S.C. § 103(a) over Wygodny et al. (U.S. Patent 6,282,701) in view of Kaler (U.S. Patent 6,467,052). Appellants respectfully traverse these rejections in light of the following remarks.

Contrary to the Examiner's assertion, Wygodny fails to teach a thread-safe remote control service which is executable on the client computer system to receive control requests from an external source to initiate and manage the debug services on the client computer system.

Wygodny teaches a method to trace an application at a remote site wherein a developer creates a preset TCI file of tracing instructions (Wygodny, column 5, lines 28-31) and supplies the TCI file along with a bugtrapper agent to a remote customer who then executes the agent and the client application and sends a generated trace file back to a developer via email for analysis (Wygodny, column 3, lines 29-39).

Specifically, Wygodny states: "Remote mode is used primarily to provide support to *users 110 that are located remotely relative to the developer 112.* In remote mode, the *agent 104 is provided to the user 110* as a stand-alone component that enables the user to generate a trace log file that represents the execution of the client." (col. 6, lines 21 – 26) Wygodny further states: "*From the perspective of the remote user, the agent 104 acts essentially as a black box* that records the execution path of the client 102. As explained above, the trace itself is not displayed on the screen, but immediately after the bug reoccurs in the application, *the user 110 can dump the trace data to the trace log file 122*

and send this file to the developer 112 (such as by email) for analysis." (col. 6, lines 38 – 45)

Figures 1A – 1C demonstrates the three basic steps for debugging a remote client under Wygodny. First, a developer creates the necessary TCI file containing information on what parts of the client application should be traced (Fig 1A). Next the developer sends the TCI file and "small tracing application called the agent to a user" (Wygodny, column 5, lines 38-39, and Fig. 1B). The user then runs the agent and the client application together on the user's computer (Wygodny, column 5, line 40).

In support of his contention that Wygodny teaches a thread-safe debugging system comprising a thread-safe remote control service which is executable on the client computer system to receive control requests from an external source, the Examiner states in the Final Office Action that Fig. 1B of Wygodny "indicates that bugtrapper agent 104 is on the Agent side (i.e. remotely located) while the traced application (item 102) is on the Client side (remote to the agent side)." Apparently the Examiner is asserting that the agent side in Fig. 1B of Wygodny is an external source for control requests to the client side to initiate and manage the debug services on the client side. However, **all objects depicted in Fig. 1B are located at the customer site (i.e. on the same computer)**, as is indicated by the underlined caption "At the customer site" located immediately above TCI file 120 in Fig. 1B. Furthermore, the "client side" refers to the *traced user application,* which is identified as the "client" in Fig. 1B and at col. 5, lines 30 – 31, while the "agent side" refers to the bugtrapper agent 104. Both the traced user application (client) and the bugtrapper (agent) run together on the customer's computer. Thus, Fig. 1B of Wygodny does not illustrate a thread-safe remote control service which is executable on the client computer system to receive control requests from an external source to initiate and manage the debug services on the client computer system.

Wygodny clearly teaches that the customer runs the agent on the same computer as the client application (Wygodny, column 16, lines 17-25). Wygodny further states,

"[t]he agent 104 and the client-side trace library 125 run in the same context ... [f]or the purposes herein, a context can be a process, a thread, or any other unit of dispatch in a computer operating system" (Wygodny, column 5, lines 54-60). Wygodny expressly teaches that both the agent side and client side run in the same process or thread in a computer system and therefore that the bugtrapper agent is not an external source to control requests to the client user application.

In regards to Fig 1B, the Examiner, in the Advisory Action dated January 29, 2004, states, "[i]f both the client side and agent side were considered the same computer, there would be no need to label each side." Appellants assert that it is well known and common practice to illustrate multiple modules, objects, processes, etc, running on the same computer using separately labeled blocks.

Appellants further submit that the Examiner's argument (from the Advisory Action) that the agent side and client side of the Wygodny's bugtrapper system reside on separate computers contradicts his previous argument (from the first Office Action) that Wygodny's bugtrapper agent is a thread-safe remote control service which is executable on the client computer system to receive control requests from an external source. The rejection is improper because the Examiner is arguing on the one hand that Wygodny's bugtrapper agent is a thread-safe remote control service which is executable on the same client computer system as the debug service, and on the other hand arguing that the bugtrapper agent is an external source not executing on the client computer system for providing control requests.

Appellants further submit that in the remote mode of Wygodny, the agent program and the client-side trace library use the information in the TCI file to determine what information is collected from a traced client program. "[T]he TCI file contains instructions to the client-side trace library 125 regarding the trace data to collect" (Wygodny, column 6, lines 3-5). Wygodny also teaches that the user executes the agent and can cause the agent to dump the trace data to a disk file that can be sent back to the

developer (Wygodny, column 5, lines 40-41, and column 6, lines 7-12). Thus, in remote mode, Wygodny teaches using a preset file of instructions to control the tracing operation. Appellants can find no language in Wygodny regarding the bugtrapper agent receiving control commands from an external source. Thus, Wygodny's bugtrapper clearly does not receive control requests from an external source to initiate and manage the debug services.

Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Wygodny clearly does not anticipate Appellants' claimed invention. Since the rejection is not supported by the teaching of the cited reference, Appellants respectfully request reversal of the Examiner's rejection of claims 1, 9-10, 13-15, 17-20, 22, 30-32, 35-37, 39-42, 44, 52-53, 56-58, and 60- 63.

B. Claims 2-4, 23-25, and 45-47

The Examiner rejected claims 2, 23, and 45 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 2, 23 and 45 is unsupported by the cited reference for at least the reasons given above in Argument A. Furthermore, contrary to the Examiner's assertion, Wygodny fails to teach a debug print function which is operable independently of the remote control service, wherein the thread-safe debug print function is operable to provide debug output for one or more threads of the multi-threaded application such that the debug output of each of the one or more threads remains distinct from the debug output of the other threads.

Wygodny teaches a BugTrapper system that can monitor the execution of a program and collect trace data (Wygodny, column 3, lines 4-10). Appellants submit that monitoring a program's execution and recording trace information does not anticipate providing an independently operable debug print function for one or more threads of the multi-threaded application.

Under Wygodny, the "agent attaches to the client by loading a client-side trace library" (Wygodny, column 5, lines 41-43). In addition to the fact that Wygodny's BugTrapper agent application is not a remote control service, as shown above in Argument A, Appellants can find no teaching in Wygodny regarding the trace output functioning independently of the bugtrapper agent application. Therefore, the rejection of claims 2-4, 23-25, and 45-47 is not supported by the teachings of the cited reference and reversal of the rejection of these claims is respectfully requested.

C. Claims 5, 26 and 48

The Examiner rejected claims 5, 26, and 48 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 5, 26, and 48 is unsupported by the cited reference for at least the reasons given above in Arguments A and B. Additionally, Wygodny fails to teach a debug print function which is operable to provide debug output for one or more threads of the multi-threaded application, wherein the debug output of each of the one or more threads may be directed to an output target and wherein the output target comprises a standard output terminal.

Wygodny clearly states, "the trace itself is not displayed on the screen, but ... the user can dump the trace data to the trace log file" (Wygodny, column 6, lines 41-44).

Wygodny also teaches that the trace log file is "written using an encoded format that is not readily decipherable by the user" (Wygodny, column 6, lines, 28-29) and that an analyzer application converts the encoded trace data back into "a source level format" and provide the developer with various debugging options (Wygodny, column 7, lines 39-53).

Appellants assert that encoded trace data, and subsequently displaying data resulting from analyzing that data does not constitute debug print function which is operable to provide debug output, wherein the debug output may be directed to an output target and wherein the output target comprises a standard output terminal. Thus, the rejection of claims 5, 26, and 48 is not supported by the teachings of the cited reference and reversal of the rejection of these claims is respectfully requested.

D. Claims 6, 27, and 49

The Examiner rejected claims 6, 27, and 49 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 6, 27, and 49 is unsupported by the cited reference for at least the reasons given above in Arguments A and B. Furthermore, Wygodny fails to teach a debug print function which is operable to provide debug output for one or more threads of the multi-threaded application, wherein the debug output of each of the one or more threads may be directed to an output target and wherein the output target comprises a recipient computer system coupled to the client computer system. **In fact, Wygodny teaches the opposite.** Under Wygodny, the trace data collected by the client-side library is written to the trace buffer. On command from the user, the agent copies the contents of the trace buffer to a trace log file. The user sends the trace log file back to the developer. (Wygodny, column 6, lines 5-12). Hence, the output of Wygodny's tracing is directed to

a trace buffer on the same computer as the client application, not a recipient computer system coupled to the client computer system.

Therefore, Wygodny fails to teach a debug print function which is operable to provide debug output, wherein the debug output may be directed to an output target and wherein the output target comprises a recipient computer system coupled to the client computer system. Since the rejection of claims 6, 27 and 49 is not supported by the teachings of the cited reference, reversal of the rejection of these claims is respectfully requested.

E. Claims 7, 28, and 50

The Examiner rejected claims 7, 28, and 50 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 7, 28, and 50 is unsupported by the cited reference for at least the reasons given above in Arguments A and B. Additionally, Wygodny fails to teach a debug print function which is operable to provide debug output for one or more threads of the multi-threaded application, wherein the debug output of each of the one or more threads may be directed to an output target and wherein the output target comprises a plurality of remote diagnostic tools.

Wygodny teaches a system where trace data collected by the client-side trace library is written to a shared buffer and optionally stored to a disk file for later analysis by an analyzer program (Wygodny, column 6, lines 7-14). Hence, the output target under Wygodny is not a plurality of diagnostic tools, but rather a memory buffer that can be both saved to disk and loaded into an analyzer program. Furthermore, Wygodny teaches of only a single analyzer application (See bugtrapper analyzer 1-6 in Figs. 1A, 1C, and 2).

Therefore, Wygodny fails to teach a debug print function which is operable to provide debug output, wherein the debug output may be directed to an output target and wherein the output target comprises a plurality of remote diagnostic tools. Since the rejection of claims 7, 28, and 50 is not supported by the teachings of the cited reference, reversal of the rejection of these claims is respectfully requested.

F. Claims 8, 29, and 51

The Examiner rejected claims 8, 29, and 51 under 35 U.S.C. § 103(a) as being unpatentable over Wygodny in view of "the applicant choice of languages used to implement his invention" (Final Office Action, dated 22-OCT-2003). Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 8, 29, and 51 is unsupported by the cited reference for at least the reasons given above in Arguments A and B. Furthermore, this rejection is clearly improper as the Examiner is attempting to reject the claims in view of Applicants' own work. On its face, the rejection is made over Wygodny in view of applicants' own teachings. It is well settled law that a claim cannot be rejected based on the applicants' own teachings.

Furthermore, the reasons stated in the first Office Action (dated May 5, 2003) referred to other programming languages, whereas claims 8, 29 and 51 refer to a plurality of regional or national languages, not programming languages. The Examiner clearly has not established a proper rejection of claims 8, 29 and 51 and therefore reversal is respectfully requested.

G. Claims 11, 33, and 54

The Examiner rejected claims 11, 33, and 54 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the

following remarks.

The rejection of claims 11, 33, and 54 is unsupported by the cited reference for at least the reasons given above in Argument A. Additionally, Wygodny fails to teach wherein the remote control service is operable to allow a remote source to switch the debug services on and off for each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects by name.

As described above, Wygodny fails to teach a remote control service. Further, Wygodny also fails to teach allowing a *remote source* to switch the debug services on and off. As described above, Wygodny teaches that a TCI file contains "instructions to the client-side trace library regarding the trace data to be collected" (Wygodny, column 6, lines 3-5). Hence, tracing is turned on or off locally according to the instructions contained in the TCI file as pre-determined by the developer.

Therefore, Wygodny fails to teach that the remote control service is operable to allow a remote source to switch the debug services on and off for each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects by name. Therefore, the rejection of claims 11, 33, and 54 is not supported by the teachings of the cited reference and reversal of the rejection of these claims is respectfully requested.

H. Claims 12, 34, and 55

The Examiner rejected claims 12, 34 and 55 under 35 U.S.C. § 102(e) as being anticipated by Wygodny. Appellants respectfully traverse this rejection in light of the following remarks.

The rejection of claims 12, 34, and 55 is unsupported by the cited reference for at least the reasons given above in Argument A. Additionally, Wygodny fails to teach

wherein the remote control service is operable to allow a remote source to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name.

As described above, Wygodny fails to teach a remote control service. Further, Wygodny also fails to teach allowing a *remote source* to switch the debug services on and off. As described above, Wygodny teaches that a TCI file contains "instructions to the client-side trace library regarding the trace data to be collected" (Wygodny, column 6, lines 3-5). Hence, tracing is turned on or off locally according to the instructions contained in the TCI file as pre-determined by the developer.

Further, Appellants can find no teaching in Wygodny regarding the use of character wildcards or other pattern techniques for identifying a set of the debug objects by name.

Therefore, Wygodny fails to teach that the remote control service is operable to allow a remote source to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name. Since the rejection of claims 12, 34 and 55 is not supported by the teachings of the cited reference, reversal of the rejection of these claims is respectfully requested.

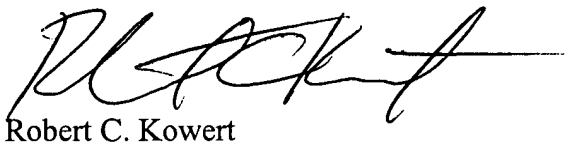
IX. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejections of claims 1 – 15, 17 – 37, 39 – 58 and 60 – 64 were erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $330.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit

Account No. 501505/5181-48500/RCK. This Appeal Brief is submitted in triplicate along with a return receipt postcard.

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: March 23, 2004

X. APPENDIX

The claims on appeal are as follows.

1. A thread-safe debugging system comprising:

a thread-safe debug service which is executable on a client computer system to provide debug services to debug a multi-threaded application which is executable on the client computer system; and

a thread-safe remote control service which is executable on the client computer system to receive control requests from an external source to initiate and manage the debug services on the client computer system after initiation of the multi-threaded application.

2. The thread-safe debugging system of claim 1, wherein the debug client comprises a thread-safe debug print function which is operable independently of the remote control service, wherein the thread-safe debug print function is operable to provide debug output for one or more threads of the multi-threaded application such that the debug output of each of the one or more threads remains distinct from the debug output of the other threads.

3. The thread-safe debugging system of claim 2, wherein the debug output of each of the one or more threads may be directed to an output target.

4. The thread-safe debugging system of claim 3, wherein the output target comprises a file.

5. The thread-safe debugging system of claim 3, wherein the output target comprises a standard output terminal.

6. The thread-safe debugging system of claim 3, wherein the output target comprises a recipient computer system coupled to the client computer system.

7. The thread-safe debugging system of claim 3, wherein the output target comprises a plurality of remote diagnostic tools.

8. The thread-safe debugging system of claim 2, wherein the thread-safe debug print function is operable to provide debug output in a plurality of regional or national languages.

9. The thread-safe debugging system of claim 1, wherein the remote control service is operable to switch the debug services on and off after initiation of the multi-threaded application.

10. The thread-safe debugging system of claim 1, wherein the debug service comprises one or more debug objects, wherein each of the one or more debug objects corresponds to a component of the multi-threaded application, wherein each of the one or more debug objects is operable to provide the debug services to the corresponding component of the multi-threaded application.

11. The thread-safe debugging system of claim 10, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects by name.

12. The thread-safe debugging system of claim 10, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name.

13. The thread-safe debugging system of claim 10, wherein the debug services comprise a service to list all the debug objects in the multi-threaded application.

14. The thread-safe debugging system of claim 10, wherein the debug services comprise a service to list a state of each debug object in the multi-threaded application.

15. The thread-safe debugging system of claim 1, wherein the debug services comprise a service to profile execution of the multi-threaded application.

17. The thread-safe debugging system of claim 1, wherein the debug services comprise a service to trace program execution through the multi-threaded application.

18. The thread-safe debugging system of claim 1, wherein the debug services comprise a service to collect run-time statistics on the execution of the multi-threaded application.

19. The thread-safe debugging system of claim 1, wherein the debug services comprise a service to log statistical information on the execution of the multi-threaded application.

20. The thread-safe debugging system of claim 1, wherein the debug services comprise a service to log performance information on the execution of the multi-threaded application.

21. The thread-safe debugging system of claim 1, wherein the external source comprises a third party application executing on a remote computer system, wherein the remote computer system is coupled to the client computer system over a network.

22. A thread-safe debugging method comprising:

initiating execution of a multi-threaded application on a client computer system, wherein the multi-threaded application is linked to a thread-safe debug service, and wherein the debug service is executable to provide debug services to debug the multi-threaded application;

initiating one of the debug services from a remote source through a thread-safe remote control service executing on the client computer system after initiating execution of the multi-threaded application on the client computer system, wherein the remote control service comprises a network protocol interface which is operable to receive control requests from the remote source; and

managing the one of the debug services from the remote source through the remote control service after initiating the debug service.

23. The thread-safe debugging method of claim 22, wherein the thread-safe debug service comprises a thread-safe debug print function which is operable independently of the remote control service, wherein the thread-safe debug print function is operable to provide debug output for one or more threads of the multi-threaded application such that the debug output of each of the one or more threads remains distinct from the debug output of the other threads.

24. The thread-safe debugging method of claim 23, wherein the debug output of each of the one or more threads may be directed to an output target.

25. The thread-safe debugging method of claim 24, wherein the output target comprises a file.

26. The thread-safe debugging method of claim 24, wherein the output target comprises a standard output terminal.

27. The thread-safe debugging method of claim 24, wherein the output target comprises a recipient computer system coupled to the client computer system.

28. The thread-safe debugging method of claim 24, wherein the output target comprises a plurality of remote diagnostic tools.

29. The thread-safe debugging method of claim 23, wherein the thread-safe debug print function is operable to provide debug output in a plurality of regional or national languages.

30. The thread-safe debugging method of claim 22, wherein the remote control service is operable to allow a remote source to switch the debug services on and off after initiation of the multi-threaded application.

31. The thread-safe debugging method of claim 22, wherein the debug service comprises one or more debug objects, wherein each of the one or more debug objects corresponds to a component of the multi-threaded application, wherein each of the one or more debug objects is operable to provide the debug services to the corresponding component of the multi-threaded application.

32. The thread-safe debugging method of claim 31, wherein the multi-threaded application is linked to a thread-safe debug service by embedding each of the one or more debug objects into the corresponding component of the multi-threaded application.

33. The thread-safe debugging method of claim 31, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for

each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects by name.

34. The thread-safe debugging method of claim 31, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name.

35. The thread-safe debugging method of claim 31, wherein the managing the one of the debug services comprises listing all the debug objects in the multi-threaded application.

36. The thread-safe debugging method of claim 31, wherein the managing the one of the debug services comprises listing a state of each debug object in the multi-threaded application.

37. The thread-safe debugging method of claim 22, wherein the managing the one of the debug services comprises profiling execution of the multi-threaded application.

39. The thread-safe debugging method of claim 22, wherein the managing the one of the debug services comprises tracing program execution through the multi-threaded application.

40. The thread-safe debugging method of claim 22, wherein the managing the one of the debug services comprises collecting run-time statistics on the execution of the multi-threaded application.

41. The thread-safe debugging method of claim 22, wherein the managing the one of the debug services comprises logging statistical information on the execution of the multi-threaded application.

42. The thread-safe debugging method of claim 22, wherein the managing the one of the debug services comprises logging performance information on the execution of the multi-threaded application.

43. The thread-safe debugging method of claim 22, wherein the external source comprises a third party application executing on a remote computer system, wherein the remote computer system is coupled to the client computer system over a network.

44. A carrier medium comprising program instructions for thread-safe debugging, wherein the program instructions are computer-executable to perform:

linking a thread-safe debug service to a multi-threaded application executable on a client computer system, wherein the thread-safe debug service comprises one or more debug services for debugging the multi-threaded application;

initiating one of the debug services from a remote source through a thread-safe remote control service executing on a server computer system after initiation of the multi-threaded application, wherein the remote control service comprises a network protocol interface which is operable to receive control requests from the remote source; and

managing the one of the debug services from the remote source through the remote control service after initiating the debug service.

45. The carrier medium of claim 44, wherein the thread-safe debug service comprises a thread-safe debug print function which is operable independently of the remote control service, wherein the thread-safe debug print function is operable to

provide debug output for one or more threads of the multi-threaded application such that the debug output of each of the one or more threads remains distinct from the debug output of the other threads.

46. The carrier medium of claim 45, wherein the debug output of each of the one or more threads may be directed to an output target.

47. The carrier medium of claim 46, wherein the output target comprises a file.

48. The carrier medium of claim 46, wherein the output target comprises a standard output terminal.

49. The carrier medium of claim 46, wherein the output target comprises a recipient computer system coupled to the client computer system.

50. The carrier medium of claim 46, wherein the output target comprises a plurality of remote diagnostic tools.

51. The carrier medium of claim 45, wherein the thread-safe debug print function is operable to provide debug output in a plurality of regional or national languages.

52. The carrier medium of claim 44, wherein the remote control service is operable to allow a remote source to switch the debug services on and off after initiation of the multi-threaded application.

53. The carrier medium of claim 44, wherein the debug service comprises one or more debug objects, wherein each of the one or more debug objects corresponds to a component of the multi-threaded application, wherein each of the one or more debug

objects is operable to provide the debug services to the corresponding component of the multi-threaded application.

54. The carrier medium of claim 53, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for each of the corresponding components of the multi-threaded application by referencing each of the corresponding debug objects by name.

55. The carrier medium of claim 53, wherein the remote control service is operable to allow a remote source to switch the debug services on and off for a set of the corresponding components of the multi-threaded application by specifying a pattern to select a set of the debug objects by name.

56. The carrier medium of claim 53, wherein the managing the one of the debug services comprises listing all the debug objects in the multi-threaded application.

57. The carrier medium of claim 53, wherein the managing the one of the debug services comprises listing a state of each debug object in the multi-threaded application.

58. The carrier medium of claim 44, wherein the managing the one of the debug services comprises profiling execution of the multi-threaded application.

60. The carrier medium of claim 44, wherein the managing the one of the debug services comprises tracing program execution through the multi-threaded application.

61. The carrier medium of claim 44, wherein the managing the one of the debug services comprises collecting run-time statistics on the execution of the multi-threaded application.

62. The carrier medium of claim 44, wherein the managing the one of the debug services comprises logging statistical information on the execution of the multi-threaded application.

63. The carrier medium of claim 44, wherein the managing the one of the debug services comprises logging performance information on the execution of the multi-threaded application.

64. The carrier medium of claim 44, wherein the external source comprises a third party application executing on a remote computer system, wherein the remote computer system is coupled to the client computer system over a network.